The Software AG installer creates the instance under the parent directory ~\IntegrationServer\instances.

Parent directory contains common and shared files that all server instances use.

Each instance has its own home directory under parent directory that contains its own packages, configuration files, log files, and updates.

~\IntegrationServer\instances\default

~\IntegrationServer\packages – Core Packages

~\IntegrationServer\updates - updates and fixes.

~\IntegrationServer\bin

~\IntegrationServer\config

~\IntegrationServer\lib

~\IntegrationServer\web

~\IntegrationServer\instances – Contains different instances

~\IntegrationServer\conf

~\IntegrationServer\docker : To facilitate running Integration Server in a Docker container.

~\IntegrationServer\features

~\IntegrationServer\sdk

~\IntegrationServer\instances\default\packages – instance packages

IS-Package: The webMethods IS hosts two type of packages

Core IS packages : provided as part of Integration Server.

Created by users : Using Designer.

Each package has its own class loader.

Service Execution Flow:

* Authenticates the client
* Use the existing session
* Look up of supplied service name
* Checks if requested HTTP method is allowed for the service
* Check access of client in ACL
* Audit the request if auditing is enabled
* Check the service-results cache or invoke the service
* Encodes the service results as specified by the content type and return the result

IS employs two kinds of class loaders

IS Server Class Loader loads the classes that comprise the core of Integration Server.

loader loads from the Server Classpath.

IS Package Class Loaders load Integration Server packages

OSGi bundle class loader : parent classloader for IS Server Class Loader

Each package has one manifest.v3 file

~\IntegrationServer\instances\default\packages\WmARTExtDC\manifest.v3

Here we decide if a package uses its own class loader or defers to the Integration Server class loader.

if the following is specified

<value name='classloader'>package</value>

Integration Server uses the package class loader

In this case the jar files (containing classes to be loaded) must be in the ~\instances\instance\_name\packages\packageName\code\jars directory.

PackageA's class loader serches for class in PackageA directories, in this order:

* PackageA\code\jars
* PackageA\code\classes
* PackageA\lib
* PackageA\resource folders

Even after searching above folders If the package class loader does not find the class the class loader moves up the dependency chain, searching packages on which PackageA depends.

By default (when class loader is not specified) class loading is defer to the parent class loader (IS Class loader)

When the Integration Server receives a request to load a class for which no package information is available,

by default, it will search all packages in the ~\instances\instance\_name\packages directory

Shared jar files used by services in multiple IS packages are stored within the same package as a service and are placed in

~\instances\instance\_name\lib\jars

If you place the file in the jar or classes directory for any package. It will be available just to that package and it's dependent package.

Place third-party jar files (JNDI provider, or custom JDBC drivers) in

~\instances\instance\_name\lib\jars\custom

Where package/IS class loader resides?

How to define package dependency?

Integration Server Does Not Know Where the Class Lives? How does we pass this info to IS